APPLICATION UNDER UNITED STATES PATENT LAWS

Atty. Dkt. No. <u>PW 282961</u>
<div align="center">(M#)</div>

Invention: ELECTING A MASTER SERVER USING ELECTION PERIODIC TIMER IN FAULT-TOLERANT DISTRIBUTED DYNAMIC NETWORK SYSTEMS

Inventor (s): Anand SRINIVASEN
Pramod DHAKAL

Pillsbury Winthrop LLP
Intellectual Property Group
1600 Tysons BoulevardMcLean, VA 22102
Tel: (703) 905-2000
Attorneys
Telephone:

<u>This is a:</u>

☐ Provisional Application

☒ Regular Utility Application

☐ Continuing Application
  ☐ The contents of the parent are incorporated by reference

☐ PCT National Phase Application

☐ Design Application

☐ Reissue Application

☐ Plant Application

☐ Substitute Specification
  <u>Sub. Spec</u> Filed _____
       in App. No. _____/_____

☐ <u>Marked up Specification re</u>
  Sub. Spec. filed _____
       In App. No _____/_____

<div align="center">

SPECIFICATION

</div>

ELECTING A MASTER SERVER USING ELECTION PERIODIC TIMER IN FAULT-TOLERANT DISTRIBUTED DYNAMIC NETWORK SYSTEMS

1.   Application Data

[0001] This application relates to and claims priority from U.S. patent Application No. 60/312,094, titled "Electing a Master Server Using Election Periodic Timer in Fault-Tolerant Distributed Dynamic Network Systems," filed August 15, 2001, the contents of which are incorporated herein by reference.

[0002]   This patent application and another are being filed simultaneously that relate to various aspects of fault tolerant distributed dynamic network systems. The other patent application is entitled "Self-Monitoring Mechanism in Fault-Tolerant Distributed Dynamic Network Systems" and has the same inventors and is commonly owned herewith. The subject matter of the application entitled "Self-Monitoring Mechanism in Fault-Tolerant Distributed Dynamic Network Systems" is hereby incorporated herein by reference.

BACKGROUND

2.   Field of the Invention

[0003]   Aspects of the present invention relate to the field of network systems. Other aspects of the present invention relate to fault-tolerant network systems.

3.   General Background and Related Art

[0004]   Client and server architecture is nowadays adopted in most computer application systems. With this architecture, a client sends a request to a server and the server processes the client's request and sends results back to the client. Typically, multiple clients may be connected to a single server. For example, an electronic commerce system

or an eBusiness system may generally comprise a server connected to a plurality of clients. In such an eBusiness system, a client may conduct business electronically by requesting the server to perform various business-related computations such as recording a particular transactionor generating a billing statement.

[0005] More and more client and server architecture based application systems cross networks. For example, a server that provides eBusiness related services may be located in California in the U.S.A. and may be linked to clients across the globe via the Internet. Such systems may be vulnerable to network failures. A problem occurring at any location along the pathways between a server and its clients may compromise the quality of the services provided by the server.

[0006] A typical solution to achieve a fault tolerant server system is to distribute replicas of a server across, for example, geographical regions. To facilitate the communication between clients and a fault tolerant server system, one of the distributed servers may be elected as a master server. Other distributed servers in this case are used as back-up servers. The master server and the back-up servers together form a virtual server or a server group.

[0007] Fig. 1 shows a configuration of a client and a server group across network. In Fig. 1, a server group comprises a master server 110 and a plurality of back-up servers 120a, …,120b, 120c, … 120d. The master server 110 communicates with its back-up servers 120a, 120b, 120c, and 120d via network 140. The network 140, which is representative of a wide range of communication networks in general such as the Internet, is depicted here as a "cloud". A client 150 in Fig. 1 communicates with the server group

via the master server 110 through the network 140, sending requests to and receiving replies from the master server 110.

[0008]   A global name server 130 shown in Fig. 1 may also be part of the configuration. The global name server 130 is where the master server 110 registers its mastership and where the reference to a server group, such as the one shown in Fig. 1, can be acquired or retrieved.   The global name server 130 may also be distributed according to, for example, geographical locations (not shown in Fig. 1).   In this case, the distributed name servers may coordinate among themselves to maintain the integrity and the consistency of the registration information.

[0009]   In Fig. 1, even though the client 150 interfaces only with the master server 110, all the back-up servers maintain the same state as the mater server 110.   That is, client requests are forwarded to all back-up servers 120a, 120b, 120c, and 120d and the back-up servers concurrently process the client requests.   The states of the back-up servers are continuously synchronized with the state of the master server 110.

[0010]   In a fault tolerant server system, when the master server fails, back-up servers may elect a new master.   The newly elected master then resumes the communications to the clients and the other back-up servers.   Fig. 2 shows such a fault tolerant system.   In Fig. 2, when the master server 110a fails, the back-up servers elect a new master server 110b.   Once elected, the new master server 110b registers its mastership with the name server and resumes the functionality of the original master server 110a.

[0011]   There are various challenges associated with electing a new master in a fault-tolerant server system.   Depending on the distribution scope of the servers from the same

server group, the degree of the difficulty varies. For example, a fault-tolerant server system distributed across the globe may have to deal with more challenging issues, compared with a fault-tolerant server system across a LAN. Furthermore, when a server group is distributed across the globe, the communication delays between the master server and different back-up servers may differ significantly. In this case, it may be more difficult to synchronize between the master and the back-up servers.

[0012] When electing a new master server, the involved servers may send messages to each other. When there are a large number of back-up servers distributed across the network, hundreds or even thousands of election messages are often sent, causing waste of resources. In addition, depending on which back-up server is elected as the new master server, the number of messages to be sent among back-up servers may vary.

SUMMARY OF THE INVENTION

[0013] This invention provides a way for a fault-tolerant server group in distributed dynamic network systems to automatically elect a master server, when an original master server is not functional, using at least one election periodic timer, each associated with one server in the server group. The election periodic timer causes the election to occur at different times for at least some of the servers.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] The present invention is further described in the detailed description which follows, by reference to the noted drawings by way of non-limiting embodiments, in which like reference numerals represent similar parts throughout the several views of the drawings, and wherein:

[0015]   Fig. 1 shows a typical configuration of a client and a server group in a fault-tolerant distributed network;

[0016]   Fig. 2 illustrates the scenario in which a back-up server is elected as a new master server when the original master server fails;

[0017]   Fig. 3 shows one embodiment of the invention, in which an election mechanism is installed on every server in a server group;

[0018]   Fig. 4 is a high level functional block diagram of an election mechanism;

[0019]   Fig. 5 is a sample flowchart of a detection unit, which detects when a master server fails;

[0020]   Fig. 6 is a high level functional block diagram of an election unit;

[0021]   Figs. 7a and 7b illustrate two sample periodicity schemes;

[0022]   Fig. 8 shows a sample flowchart of a process, in which a server declares itself as a new master; and

[0023]   Fig. 9 is a sample flowchart of a process, in which new mastership is being established.

DETAILED DESCRIPTION OF SEVERAL EMBODIMENTS

[0024]   The invention is described below, with reference to detailed illustrative embodiments. It will be apparent that the invention can be embodied in a wide variety of

forms, some of which may be quite different from those of the disclosed embodiments. Consequently, the specific structural and functional details disclosed herein are merely representative and do not limit the scope of the invention.

[0025]   The processing described below may be performed by a general-purpose computer alone or in connection with a specialized computer. Such processing may be performed by a single platform or by a distributed processing platform. In addition, such processing and functionality can be implemented in the form of special purpose hardware or in the form of software being run by a general-purpose computer. In this application, the term "mechanism" is termed to refer to any such implementation. Any data handled in such processing or created as a result of such processing can be stored in any memory as is conventional in the art. By way of example, such data may be stored in a temporary memory, such as in the RAM of a given computer system or subsystem. In addition, or in the alternative, such data may be stored in longer-term storage devices, for example, magnetic disks, rewritable optical disks, and so on. For purposes of the disclosure herein, a computer-readable media may comprise any form of data storage mechanism, including such existing memory technologies as well as hardware or circuit representations of such structures and of such data.

[0026]   Fig. 3 shows an embodiment 300 of the invention, in which an election mechanism is installed and may be activated on all the servers, including a master server and a plurality of back-up servers, that form a server group.   System 300 comprises a server group 320 which includes a master server 110, a plurality of back-up servers 1-N 120a,...,120b,120c,...,120d,   and   a   plurality   of   election   mechanisms 310a,..,310b,310c,...,310d (attached to the master server 110 as well as to the back-up

servers 1-N 120a,...,120b,120c,...,120d), a name server 130, a client 150, and a network 140.

[0027]   The master server 110 and the back-up servers 1-N 120a,...,120b,120c,...,120d form the fault tolerant server group 320 that provides the client 150 services. Example of such services may include Internet Service Provider services or on-line shopping services. The servers in the server group 320 may be distributed across the globe. For example, the master server 110 may be physically located in Ottawan, Canada, the back-up server 1 120a may be physically located in Atlanta, USA, the back-up server i 120b may be physically located in Bangalore, the back-up server j 120c may be physically located in Sydney, and the back-up server N 120d may be physically located in Tokyo. The servers in the server group 320 communicate with each other via the network 140 which is representative of a wide range of communications networks in general.

[0028]   The client 150 communicates with the server group 320 by interfacing with the master server 110 through the network 140. The master server 110 interacts with the back-up servers via the network 140. When the client 150 sends a request to the master server 110, the master server 110 forwards the client's request to the back-up servers 1-N (120a,...,120b,120c,...,120d). All the servers in the server group 320 concurrently process the client's request and the master server 110 sends the results back to the client 150. The states of the servers in the server group 320, including the master server 110 and the back-up servers 1-N 120a,...,120b,120c,...,120d, are continuously synchronized.

[0029]   The mastership of the master server 110 may be registered in the name server 130. Each server group may register a desired number of servers as the master servers and the registration may explicitly use both the identification of the server group as well as

the identification of the master server being registered. Through the name server 130, a client may access or retrieve information such as registered master servers.

[0030]   The name server 130 may also be distributed (not shown in Fig. 3). In this case, the integrity and the consistency of the registrations for the master servers may be maintained across the distributed name servers. For instance, if distributed name servers have multiple copies of the registrations, these copies should contain the same content. Also when the mastership for a server group changes, the copies of the original registration, which may be scattered in distributed name servers, may have to be updated simultaneously to maintain the consistency of the registration information.

[0031]   As shown in Fig. 3, the election mechanisms 310a,...,310b,310c,...,310d are attached to all the servers in the server group 320. With an election mechanism running, a back-up server may regularly check to see whether the master server 110 is functional. When the master server 110 is detected to be not functioning, the election mechanism running on each back-up server enables the underlying back-up server to participate in an election, in which a new master server is elected to replace the master server 110. The detection performed by each back-up server to see whether the master server 110 is functional may be carried out regularly with a well defined time interval. Different back-up servers may perform the regular checks either synchronously or asynchronously.

[0032]   Fig. 4 shows a high level functional diagram of an election mechanism (310a). In Fig. 4, the election mechanism 310a comprises a detection mechanism 410 and a new master election mechanism 420. The detection mechanism 410 detects when the master server 110 fails. The detection unit 410 may be activated regularly by a timer after a specified length of time. The length of the time specified in the timer determines the

periodicity of the detection mechanism 410. The election mechanisms installed on different back-up servers may employ either equal or different periodicity for the detection. The activation may be synchronous or asynchronous.

[0033]   Figs. 7a and 7b show timers with different types of periodicity. As illustrated in Fig. 7a, timers with an equal periodicity are controlled by cycles of the same length of time. As illustrated in Fig. 7b, timers with non-equal periodicity are controlled by cycles of different lengths of time.

[0034]   When the detection mechanism 410 on a particular back-up server is activated or triggered, it sends an inquiring message to the master server 110 to check whether the master server 110 is still functional. This may be achieved by detecting whether the master server 110 responds to the inquiring message in a specified time limit. Upon detecting the failure of the master server 110, the new master election mechanism 420 enables the underlying back-up server to start an election in which a new master for the server group 320 is selected to replace the failed master server 110.

[0035]   Fig. 5 is a sample flowchart of the detection mechanism 410. In Fig. 5, the detection of the failure of an existing master server is activated by a detection periodic timer 510. The detection periodic timer 510 regulates how often each server checks whether its corresponding master server is functional. The detection periodic timer 510 may specify a particular time interval by which the detection mechanism 410 is regularly activated.

[0036]   Once the detection mechanism 410 is activated, it sends an inquiry message, at act 520, from the underlying back-up server (on which the detection mechanism 410 is

running) to the master server 110. A time-out condition may then be immediately initialized, at act 530, to start a different timer (not shown) that counts towards a time-out criterion. The time-out criterion may specify the length in time by which the underlying back-up server expects the master server 110 to respond.

[0037] If a reply to the inquiry message is received from the master server 110, determined at act 540, it indicates that the master server 110 is functional. If no reply to the inquiry is received from the master server 110, the time-out condition is evaluated at act 550. If the time-out criterion is not yet satisfied at act 550, the detection unit 410 returns to act 540 to wait for a reply from the master server 110. If the time-out criterion is satisfied at act 550, it indicates that the underlying back-up server did not receive a reply from the master server 110 within specified time-out limit. In this case, the master server 110 is considered no longer functional. In this case, the underlying back-up server enters an election process (performed by the new master election mechanism 420) via C.

[0038] If the master server 110 is functional, determined at act 540, the detection mechanism 410 may further examine, at act 560, to see whether a message from a different back-up server is received over the network. If no message is received, the detection mechanism 410 goes back to detection periodic timer 510 to wait until the detection periodic timer 510 activates it again.

[0039] If a message from a different back-up server is received at act 560, it may indicate that multiple servers have been set as masters. The detection mechanism 410 proceeds to an election process (performed by the election mechanism 420) via B. This scenario (to enter election after a back-up server detects that the master server 110 is functional) is possible because although some of the back-up servers consider the master

server 110 to be functional, there may be other back-up servers that may detect that the master server is no longer functional. For example, if some back-up servers have lost connection with the master server 110 (e.g., due to, for example, a network partition), those back-up servers may decide to elect a new master. In this case, the message received at act 560 by the underlying back-up server may include a message from a particular back-up server that claims a new mastership. In this situation, the message received at act 560 may request the underlying back-up server to accept the new mastership and to update its states accordingly.

[0040]   Referring back to Fig. 4, in the election mechanism 310a, when the master server 110 fails, detected by the detection mechanism 410, the election mechanism 310a enters an election process and the election mechanism 420 is activated. Fig. 6 shows a high level functional block diagram of the new master election mechanism 420. In Fig. 6, the new master election mechanism 420 comprises an election periodic timer 610, a master selection mechanism 620, and a mastership updating mechanism 630. The election periodic timer 610 is used to control the timing of claiming mastership. The master selection mechanism 620 is invoked when a back-up server claims itself as the newly elected master server. In this case, the newly elected master server may send a claiming message to all other servers in the same server group to establish its mastership.

[0041]   The mastership updating mechanism 630 is invoked when a back-up server receives a message that attempts to establish a new mastership. In this case, the back-up server (that receives the message) determines whether to accept or to contest the newly elected master server. In the latter case, the back-up server may send a different message

to all the servers in the same server group to revoke the newly claimed mastership. At the same time, the back-up server claims itself as the new master server.

[0042] In Fig. 6, the election periodic timer 610 may be set up in a manner that it plays a similar role as a time-out mechanism. For example, when the master server 110 is detected no longer functional, a back-up server enters an election process by first initiating the election periodic timer 610. The back-up server may then wait (or enter a sleep mode) until the time, specified by the election periodic timer 610, has elapsed and then declares itself as the new master. This wait time represents an election delay time. The length of the election delay time set up in each election periodic timer may be adjusted accordingly for the corresponding back-up server.

[0043] Figs. 7a and 7b illustrate two methods to set up the length of election delay time in the election periodic timer 610. In Figs. 7a and 7b, assume servers in a server group are denoted by $S_0, S_1, ..., S_N$, where S_0 is assumed to be a master server and $N+1$ is the total number of servers in the server group. Fig. 7a corresponds to an equally periodic timer in which the same election delay time, for example T, is set up for every server. That is, each server may wait the same amount of time before it declares its mastership. Since the master server does not participate the election, the wait time does not apply. In implementation, the wait time for a master server may simply be set as zero.

[0044] With an equal periodicity, the situation may arise in which multiple servers may claim the mastership at substantially the same time. To reach a state with only one elected master server, it may take multiple rounds of messages to settle among back-up servers. While it may be an adequate solution when the number of servers in a server group is

reasonably small, with a larger number of servers, hundreds or thousands of messages may be sent across the network that may cause inefficiency.

[0045]   In an embodiment shown in Fig. 7b, a different election delay time may be set up for each server. One mechanism for setting different election delay times for different servers is illustrated in Fig. 7b. The election delay time for a particular server, for example S_i, may be determined according to its relative rank, determined by its index value i, in the server group. The rank of a server is the inverse of its index value. Therefore, server S_i ranks higher than server S_j if $i < j$.

[0046]   Assume D_i is the election delay time of the ith server S_i. In the example shown in Fig. 7b, there are five servers ranked in the order of $(S_0, S_1, S_2, S_3, S_4)$, where server S_0 is a master server. Since a master server does not participate election, its election delay time does not apply. This may be realized by, for example, setting the election delay time for S_0 (i.e., the master server) to be zero. That is, $D_0 = 0$. Each of the four back-up servers (i.e., S_1, S_2, S_3, S_4) corresponds to a timer with different election delay times (D_1, D_2, D_3, D_4), the election delay time D_i for back-up server S_i may be defined as a summation of a base election delay time T_1 and an adjusted election delay time $\sum_{j=0}^{i-1} \delta_j$, where each $\delta_j, 0 \leq j \leq i-1$, corresponds to an adjusted waiting time for server S_j. That is, $D_i = T_1 + \sum_{j=0}^{i-1} \delta_j$, $1 \leq i \leq N$. Similarly, the adjusted waiting time for the master server may be set to zero $\delta_0 = 0$.

[0047]   With the above definition for election delay time, the base election delay time T_1 may be viewed as the minimum delay before a back-up server can start an election. Each term $\delta_j, 0 \leq j \leq i-1$, in Fig. 7b, may be defined as the maximum communication delay between server S_j and servers $S_k, j+1 \leq k \leq N$, defined as $\delta_j = \max\{C_{j,k}, j+1 \leq k \leq N\}$, where $C_{j,k}$ is the communication delay between servers S_j and S_k.

[0048]   The above defined sample periodic timer may be termed as an integral periodic timer, by which each server derives an election delay time that is based on an accumulative delay time and that is correlated with the rank of the server in its server group. With such a scheme of computing the election delay time, the lower the rank of a server is, the longer its election delay time may be because a lower rank server accumulates more communication delays. Since the server that has the shortest election delay time declares to be the master server first, the rank of a server may play a crucial role in the election.

[0049]   The rank of a server may be determined according to certain criterion. For example, it may be related to the computation power or the bandwidth capacity of the server. In this case, the back-up server that first declares its mastership may correspond to a more powerful back-up server in terms of the criterion. The rank of the servers in a server group may be set up off-line or may be re-ranked when the system configuration changes. Such configuration changes may include the replacement of servers (e.g., a new powerful server to replace an existing server) or upgrades of existing servers (e.g., more processors are added to an existing server so that its computation power is improved).

[0050]   The criterion used to rank servers may be determined based on application needs. For example, if a server group provides services mainly in scientific computation, the computation powers of the servers may determine their rank. If a different server group provides mainly real-time communication capabilities to users (e.g., video-conferencing over the Internet), the computation power of each server may become less important. In this case, the bandwidth capacity of a server may be employed to improve the quality of service for real-time video-conferencing sessions. It is also possible that the services a server group offers change with time so that the criterion used in ranking the servers may also have to be adjusted accordingly to fit what is required to support the changing services.

[0051]   Fig. 8 is a sample flowchart for the master selection mechanism 620. The election mechanism 310a running on a back-up server invokes the master selection mechanism 620 when the back-up server 120a detects that the master server 110 failed to respond to its inquiry message. The back-up server 120a enters an election process.

[0052] · The master selection mechanism 620 first sets, at act 810, the state of the underlying back-up server to a waiting state WaitElection. Assume that the underlying back-up server is the ith server or S_i. The election periodic timer for the underlying back-up server S_i is then initialized at act 820. It is set as an integral periodic timer with election delay time D_i.

[0053]   In this embodiment, the back-up server S_i waits, once enters the election process, until the elapse of election delay time D_i and then declares itself as the new master. During the waiting, the master selection mechanism 620 checks, at act 830,

whether the election delay time D_i has elapsed. If it has, the master selection mechanism 620 sets, at act 850, the state of the back-up server S_i as master and then sends out a message, at act 860, to all the servers in the server group 320.

[0054]   The message sent at act 860 informs other servers that server S_i is taking over the mastership. The message may be designated as a special message such as Declare_Master and it may carry parameters that notify the receivers who sends the message or who is taking over the mastership. In this case, the index value i of the server may suffice.

[0055]   After the Declare_Master(i) is sent out, the new mastership declared by server S_i may be contested or challenged. When this happens, the server that decides to override the new mastership declared by server S_i may send a different message to all the servers in the server group 320, attempting to revoke the mastership declared by server S_i. Therefore, after Declare_Master(I) message is sent out, the master selection mechanism 620 checks, at act 870, whether a message is received. If no message is received, the S_i's mastership is considered to be accepted and the process proceeds, via A, to the detection periodic timer 510 (Fig. 5) so that the timer for the regular checks on the new master may be reset. If a message is received at act 870, the process proceeds to the mastership updating mechanism 630 (via B).

[0056]   While the back-up server S_i is waiting for the elapse of its election delay time D_i, a different back-up server that has a shorter election delay time D_j, $j \neq i$, may take over the mastership and declare so by sending a message Declare_Master(j) to all the other

servers in the server group 320. Therefore, in Fig. 8, prior to the elapse of D_i, it is also examined, at act 840, to see whether the back-up server S_i receives a message. If no message is received, the process returns to act 830 to continue the check on the election delay time. If a message is received, the election mechanism 420 proceeds to the mastership updating mechanism 630 (via B).

[0057] Fig. 9 shows an exemplary flowchart of the mastership updating mechanism 630. At act 905, a message received by server S_i is analyzed. If the received message is an Declare_Master(idx) message sent from server S_{idx}, determined at act 910, the current status of the receiving server (S_i) is further determined. If the current status of server S_i, checked at act 915, is WaitElection (i.e., server S_i is still waiting for the elapse of its election delay time D_i and server S_i is currently a back-up server), server S_i may simply acknowledge the new mastership declared by server S_{idx}. This is achieved by setting the master state (at act 920) of server S_i to be server S_{idx}, and then terminating, at act 925, its own election delay time D_i. The process then returns to act 870 (Fig. 8) to intercept a message (via D). The return to act 870 (to wait for a message) may be necessary because some server may (different from server S_i) contest the mastership declared by server S_{idx} and may soon notify, via a message, all servers, including those that have accepted the mastership declared by server S_{idx}, to revoke such mastership.

[0058] If the current status of the receiving server (S_i) is not WaitElection, then either S_i is the original master server (that is, considered by at least some back-up servers no longer functional) or S_i is a server that is originally a back-up server and just passed its

election delay time and just set its state as the new master. That is, at this time instance, server S_i considers itself as the master yet just receives a message that declares some other server to be the new master. In this case, server S_i competes for the mastership with server S_{idx}.

[0059]   In a competing situation, different criteria may be used to determine a winner. For example, based on application needs, a server with a faster computation speed may be chosen as the winner. As another example, if the server group 320 is to provide real-time video conferencing capability to client 150 (an application that requires high bandwidth), it may be more reasonable to choose a competing server that has higher bandwidth capacity to be the new master.

[0060]   Fig. 9 presents an embodiment in which the server that has a smaller index value becomes the new master. With this embodiment, the index value may be designed to incorporate the selection criteria that are in accordance with particular application needs. For example, the index value of a server may be defined as inversely proportional to the bandwidth capacity of the server. In this way, by choosing a server with a smaller index value, the selected server can better handle the tasks that require higher bandwidth. The use of a smaller server index value is illustrated at acts 930 through 960. It should be appreciated that different embodiments may also used for the selection of a new master.

[0061]   If server S_i has a smaller index value than server S_{idx} ($i < idx$), determined at act 930, server S_i becomes the new master. In this case, the state of server S_i is set to be master at act 950 and a revoke message, Revoke_Master(idx,i), is sent, at act 960, to all the other servers in the server group 320 to notify them to replace the mastership declared by

server S_{idx} with the mastership of server S_i. The process then returns to act 870, via D, to intercept a message.

[0062] The return to act 870 (to wait for a message) may be necessary because some server, upon receiving the message Revoke_Master(idx,i), may further contest the mastership declared by server S_i and may soon notify, via a message, all servers to revoke the mastership declared by server S_i. A different embodiment is also possible in which returning back to act 870 may be avoided. If server S_i can determine, at act 930, that its index value is the smallest among all servers that are functional (instead of smaller than idx), it is not possible for anyone to contest the mastership declared by server S_i. For example, if server S_i has access to a table of IDs for all the functional servers in the same server group, it may be able to determine that it is the best (instead of better) choice to revoke the mastership declared by server S_{idx}. In this case, the process may proceed, via A, to reset the detection periodic timer 510 (not shown in Fig. 9).

[0063] If server S_{idx} has a smaller index value than server S_i ($i \geq idx$), determined at act 930, the mastership declared by server S_{idx} through message Declare_Master(idx) is accepted by server S_i. In this case, the state of server S_i is set, at act 935, to be a back-up server and the master of server S_i is set, at act 940, to be server S_{idx}. The process then returns to act 870, via D, to intercept a message. Similarly, the return to further intercept a message may be necessary because some other server may contest and may try to revoke the mastership declared by server S_{idx} by requesting all servers, via a message, to replace the mastership.

[0064]   If a received message is not Declare_Master, determined at act 910, it is further examined to see, at act 970, whether it is a Revoke_Master message. A Revoke_Master message may carry two parameters representing two server indices, for example, "reject" and "idx". The carried indices may intend to notify receivers to replace the mastership declared by a server represented by index "reject" with the mastership declared by a different server represented by index "idx".

[0065]   In the illustrated embodiment shown in Fig. 9, when server S_i receives a Revoke_Master message, instead of simply accepting the new mastership declared by server S_{idx}, server S_i may determine whether S_i itself is a more appropriate candidate for mastership by comparing its own index value with the index value of server S_{idx}.

[0066]   When the index value of server S_i is smaller than the index value of server S_{idx}, determined at act 930, server S_i declares itself as the new master by setting its own state to be the master (at act 950) and by requesting other servers, at act 960, to revoke the mastership of server S_{idx} and accepting server S_i as the master.

[0067]   When the index value of server S_i is not smaller than the index value of server S_{idx}, server S_i accepts server S_{idx} as the master. This is achieved by setting its own state to be a back-up server (at act 935) and then setting its master to be server S_{idx}.

[0068]   As discussed earlier, the illustrative details of one embodiment described above about the election mechanism 310a use integral periodic timers so that each different server enters the election process with a different election delay time. Such an election mechanism corresponds to an linear computational complexity of O(N) in terms of the

number of messages sent to complete the election, where N is the number of servers in a server group. When there are a large number of servers in a server group, an election mechanism that uses integral periodic timers for different servers may be able to limit the computational complexity of the election process.

[0069]   While the invention has been described with reference to the certain illustrated embodiments, the words that have been used herein are words of description, rather than words of limitation. Changes may be made, within the purview of the appended claims, without departing from the scope and spirit of the invention in its aspects. Although the invention has been described herein with reference to particular structures, acts, and materials, the invention is not to be limited to the particulars disclosed, but rather extends to all equivalent structures, acts, and, materials, such as are within the scope of the appended claims.